

Département Informatique

DUT Informatique
Module Système S4
2008 / 2009

Carte de Référence C

Structure Générale d'un Programme

```
                          // Incorporation des bibliotheques :
#include <stdio.h>        // - cas d'une bibliotheque standard
#include "mabib.h"        // - cas d'une bibliotheque utilisateur

float pi = 3.14 ;         // Declaration des variables globales

float aire(float r) {     // Definition des fonctions
  float surface ;         // Declaration des variables locales
  surface = 2*pi*r ;      // Instructions...
  return surface ;
}

int main() {              // Routine principale
  float rayon, s ;        // Declaration des variables locales
  printf("Rayon?\n") ;    // Instructions...
  scanf("%f",&rayon) ;
  s = aire(rayon) ;
  printf("%f\n",s) ;
  return 0 ;
}

// Une ligne de commentaires
/* Plusieurs lignes
   de commentaires */
```

Arguments de `main`

```
int main(int argc, char *argv[])
```

- `argc` est le nombre d'arguments passés en ligne de commande, nom du programme compris,
- `argv` est un tableau de chaînes de caractères, chacune représentant un argument, `argv[0]` est le nom du programme.

Préprocesseur C

Incorporation de bibliothèque	`#include` <*nom_fichier*>
Incorporation de fichier utilisateur	`#include "`*nom_fichier*`"`
Remplacement de texte	`#define` *nom texte*
Remplacement de macro	`#define` *nom(var) texte*
➥ exemple : `#define modulo(A,B) ((A)%(B))`	
Arrêt de définition	`#undef` *nom*
Exécution conditionnelle	`#if`, `#else`, `#elif`, `#endif`
nom est-il défini, ou non défini ?	`#ifdef` *nom*, `#ifndef` *nom*
Caractère de continuation à la ligne	`\`

© 2007 Joseph H. Silverman and Cédric Bastoul Permissions on back. v1.1f

Types de Données, Déclaration, Initialisation, Taille

Caractère (1 octet)	`char`
Réel (simple et double précision)	`float`, `double`
Entier	`int`
Entier court (16 bits)	`short`
Entier long (32 bits)	`long`
Entier double (64 bits)	`long long`
Sans valeur	`void`
Structure	`struct` *tag*`{`*déclarations*`}` ;
Pointeur sur `int`, `float`,...	`int *`, `float *`,...
Modifieur pour nombres positifs ou négatifs	`signed`
Modifieur pour nombres non-négatifs	`unsigned`
Modifieur pour nombres constants	`const`
Modifieur pour variables externes	`extern`
Modifieur pour constantes locales	`static`
Attribuer un nouveau nom à un type	`typedef` *type nom* ;
Déclaration d'une variable	*modifieur type nom* ;
Initialiser une variable	*type nom*=*valeur* ;
Initialiser un tableau	*type nom*`[]={`val_1`,...}` ;
Initialiser une chaîne de caractères	`char` *nom*`[]="chaine"` ;
Taille d'un élément *nom* (type `size_t`)	`sizeof(`*nom*`)`
Taille d'un type *type* (type `size_t`)	`sizeof(`*type*`)`

Constantes

Suffixe : `long`, `unsigned`, `float`	`65536L, -1U, 3.14F`
Forme exponentielle	`2.3e1`
Constante caractère	`'c'`
Retour à la ligne, tabulation, backspace	`\n, \t, \b`
Caractères spéciaux dans les chaînes	`\\, \?, \', \"`
Chaîne constante (finit par `'\0'`)	`"abc...yz"`

Pointeurs, Tableaux et Structures

Déclarer un pointeur sur *type*	*type* `*`*nom* ;
Fonction retournant un pointeur sur *type*	*type* `*f()` ;
Type pointeur générique	`void *`
Constante pointeur nul	`NULL`
Élément pointé par *pointeur*	`*`*pointeur*
Adresse de l'élément *nom*	`&`*nom*
Tableau	*nom*`[`*dim*`]`
Tableau multidimensionnel	*nom*`[`dim_1`][`dim_2`]`...
Définition d'une structure	`struct` *tag*`{`*déclarations*`}` ;
Déclaration d'un élément structure	`struct` *tag name* ;
Accès au champ d'une structure	*nom*`.`*champ*
Accès au champ d'une structure pointée 1	`(*`*pointeur*`).`*champ*
Accès au champ d'une structure pointée 2	*pointeur*`->`*champ*

Opérateurs

La priorité `1` est la plus haute. En cas d'égalité, les opérateurs unaires, l'opérateur conditionnel et les opérateurs d'affectation se règlent (*«associativité»*) de droite à gauche, tous les autres de gauche à droite.

Opérateurs Arithmétiques

Opérateur	Usage	Description	Priorité
`+`	`x+y`	ajoute `x` et `y`	4
`-`	`x-y`	soustrait `y` à `x`	4
`*`	`x*y`	multiplie `y` et `x`	3
`/`	`x/y`	divise `x` par `y`	3
`%`	`x%y`	reste de la division entière `x/y`	3
`++`	`x++`	incrémente `x` de `1`, l'évaluation de `x` a lieu **avant** incrémentation	2
`++`	`++x`	incrémente `x` de `1`, l'évaluation de `x` a lieu **après** incrémentation	2
`--`	`x--`	décrémente `x` de `1`, l'évaluation de `x` a lieu **avant** décrémentation	2
`--`	`--x`	décrémente `x` de `1`, l'évaluation de `x` a lieu **après** décrémentation	2
`-`	`-x`	opposé de `x`	2

Opérateurs Relationnels

Opérateur	Usage	Renvoie `vrai` si	Priorité
`>`	`x>y`	`x` est strictement supérieur à `y`	6
`>=`	`x>=y`	`x` est supérieur ou égal à `y`	6
`<`	`x<y`	`x` est strictement inférieur à `y`	6
`<=`	`x<=y`	`x` est inférieur ou égal à `y`	6
`==`	`x==y`	`x` et `y` sont égaux	7
`!=`	`x!=y`	`x` et `y` ne sont pas égaux	7
`&&`	`x&&y`	`x` et `y` sont vrais (`y` n'est pas forcément évalué)	11
`||`	`x||y`	`x` ou `y` sont vrais (`y` n'est pas forcément évalué)	12
`!`	`!x`	`x` est faux	2

Opérateurs Logiques Binaires

Opérateur	Usage	Opération	Priorité
`>>`	`x>>y`	décalage binaire de `x` vers la droite de `y` bits	5
`<<`	`x<<y`	décalage binaire de `x` vers la gauche de `y` bits	5
`&`	`x&y`	ET logique bit à bit entre `x` et `y`	8
`|`	`x|y`	OU logique bit à bit entre `x` et `y`	10
`^`	`x^y`	XOU (OU exclusif) logique bit à bit entre `x` et `y`	9
`~`	`~x`	complément bit à bit de `x`	2

Opérateurs (suite)

Opérateurs d'Affectation

Opérateur	Usage	Description	Priorité
=	x=y	affecte à x la valeur de y	14
+=	x+=y	x = x + y	14
-=	x-=y	x = x - y	14
*=	x*=y	x = x * y	14
/=	x/=y	x = x / y	14
%=	x%=y	x = x % y	14
>>=	x>>=y	x = x >> y	14
<<=	x<<=y	x = x << y	14
&=	x&=y	x = x & y	14
|=	x|=y	x = x | y	14
^=	x^=y	x = x ^y	14

Opérateurs de Référence, Cast et Conditionnel

Opérateur	Usage	Description	Priorité
[]	a[i]	i^{ème} élément du tableau a	1
.	s.c	champ c de la structure s	1
->	p->c	champ c de la struct. pointée par p	1
*	*p	élément pointé par p	2
&	&x	adresse de l'élément x	2
()	(t)e	(«*cast*») considère l'expression e comme étant du type t	2
? :	e?a:b	(«*conditionnel*») exécute a si l'expression e est vraie, et b sinon	13

Flot de Contrôle

Séparateur d'instructions	;
Délimiteur de bloc d'instructions	{ }
Sortie de for, while, do, cas de switch	break ;
Aller à l'itération suivante de for, while, do	continue ;
Aller à *label*	goto *label* ;
Placer un label	*label*:instruction
Retourner une valeur depuis une fonction	return *expression*

Constructions de Contrôle

Conditionnelle if
```
if (expression)
  instruction_ou_bloc1
else
  instruction_ou_bloc2
```

Conditionnelle switch
```
switch (expression) {
  case constante1: instructions1 break ;
  case constante2: instructions2 break ;
  /* ... */
  default: instructionsn
}
```

Flot de Contrôle (suite)

Boucle for
```
for (expr_init ; expr_term ; expr_incr)
  instruction_ou_bloc
```

Boucle while
```
while (expression)
  instruction_ou_bloc
```

Boucle do
```
do
  instruction_ou_bloc
while (expression) ;
```

Principales Fonctions Utilitaires Standard <stdlib.h>

Valeur absolue de l'entier n	abs(n)
Entier pseudo-aléatoire [0,RAND_MAX]	rand()
Fixer la graine de génération aléatoire à n	srand(n)
Terminer l'exécution du programme	exit(statut)
Envoyer la chaîne s au système pour exécution	system(s)
Conversions	
Convertir une chaîne s en entier	atoi(s)
Convertir une chaîne s en réel	atof(s)
Convertir le préfixe de s en réel double	strtod(s,&fin_p)
Convertir le préfixe de s (base b) en long	strtol(s,&fin_p,b)
Gestion de la Mémoire	
Allouer une zone mémoire (renvoie son adresse)	ptr=malloc(size) ;
Changer la taille d'une zone allouée	newptr=realloc(ptr,size) ;
Libérer une zone mémoire allouée	free(ptr) ;

Principales Fonctions sur les Chaînes <string.h>

s est une chaîne, cs et ct sont des chaînes pouvant être constantes

Longueur de s	strlen(s)
Copie ct dans s	strcpy(s,ct)
Concaténer ct après s	strcat(s,ct)
Comparer cs à ct (0: égaux, <0: inf, >0: sup)	strcmp(cs,ct)
Comparer les n premiers caractères de cs et ct	strncmp(cs,ct,n)
Copier n caractères de ct dans s	memcpy(s,ct,n)
Placer le caractère c dans les n premiers de s	memset(s,c,n)

Principales Fonctions Mathématiques <math.h>

Fonctions de trigonométrie	sin(x), cos(x), tan(x)
Fonctions inverses de trigonométrie	asin(x), acos(x), atan(x)
Exponentielles et logarithmes	exp(x), log(x), log10(x)
Puissance, racine carrée	pow(x,y), sqrt(x)
Arrondis (à l'entier supérieur, inférieur)	ceil(x), floor(x)

February 2007 v1.1f. Copyright © 2007 Joseph H. Silverman and Cédric Bastoul
Permission is granted to make and distribute copies of this card provided the copyright notice and this permission notice are preserved on all copies.
Send comments and corrections to Cédric Bastoul, cedric.bastoul@lri.fr and find the latest version at http://www.lri.fr/~bastoul/systeme.

Entrées / Sorties <stdio.h>

Entrée / Sortie Standard	
Flot d'entrée standard	stdin
Flot de sortie standard	stdout
Flot d'erreur standard	stderr
Écrire un caractère c	putchar(c)
Lire un caractère	getchar()
Écrire une chaîne s	puts(s)
Lire une chaîne à placer dans s	gets(s)
Écrire des informations formatées	printf("*format*",arg_1,...)
Lire des informations formatées	scanf("*format*",&arg_1,...)
Écrire dans une chaîne s	sprintf(s,"*format*",arg_1,...)
Lire depuis une chaîne s	sscanf(s,"*format*",&arg_1,...)
Entrée / Sortie sur Fichiers	
Déclaration d'un pointeur sur fichier	FILE *pf
Pointer (ouvrir) un fichier *nomfic*	fopen("*nomfic*","*mode*")
➥ modes : r (read), w (write), a (append), b (binary)	
Écrire un caractère c	putc(c,pf)
Lire un caractère	getc(pf)
Écrire une chaîne s	fputs(s,pf)
Lire une ligne (<max chars) vers s	fgets(s,max,pf)
Écrire dans un fichier	fprintf(pf,"*format*",arg_1,...)
Lire depuis un fichier	fscanf(pf,"*format*",&arg_1,...)
Écrire n éléments pointés par ptr	fwrite(ptr,eltsize,n,pf)
Lire n éléments à placer à ptr	fread(ptr,eltsize,n,pf)
Fin de fichier (type int)	EOF
Non nul si EOF déjà atteinte	feof(pf)
Fermer le fichier	fclose(pf)

Codes pour Informations Formatées

Forme générale : %[*flags*][*largeur*][.*précision*][*longueur*]*type*

➥ Les éléments avec crochets sont optionnels

[*flags*]	-	Justifier à gauche
	+	Afficher le signe pour les nombres
	espace	Afficher un espace si pas de signe
[*largeur*]	*n*	Nombre *n* de caractères minimum à afficher
[.*précision*]	.*m*	Nombre *m* maximal de décimales
[*longueur*]	h	Modifieur pour type entier short
	l	Modifieur pour type entier long
	ll	Modifieur pour type entier long long
type	c	Caractère
	d, i	Entier signé d ou i au choix
	u	Entier non signé
	f, e	Réel, notation classique f, exponentielle e
	g	Double (la notation dépend de la valeur)
	s	Chaîne de caractères
	p	Pointeur (adresse)
	%	Pour afficher le symbole % (pas d'options)